Remarks

Support for the amended claims

The amendment of claim 1 adopts a suggestion of Examiner's for overcoming the rejection under 35 U.S.C. 101; as indicated by Examiner in his suggestion, the claim as amended is fully supported by FIG. 1 and at Applicants' [0014]. The amendment of claim 6 also adopts a suggestion of Examiner's with regard to the 101 rejection; as amended, the claim makes it absolutely clear that the claimed "improved class loader" is executed "by the host computer system". The remaining amendments to claim 6 correct an inaccuracy in the claim language: as set forth in the Specification at least at [0100], the term "extend" is applied in the Specification to the class loader, not to the class being loaded by the class loader. The amendment further brings the language of claim 6 into conformity with the other claims, none of which employs the term "extend" with regard to the class being loaded.

The rejection of claims 1 and 6 under 35 U.S.C. 101

As just noted, the amendments to these claims adopt suggestions by Examiner for overcoming the rejections under 35 U.S.C. 101; since independent claims 1 and 6 as amended are now directed to statutory subject matter, so are claims 2-5 dependent from amended claim 1 and claims 7-13 dependent from amended claim 6.

Traversal of the rejection of claims 6-21 under 35 U.S.C. 103

What Applicants are claiming

The limitations of claim 6 are exemplary for those of independent claims 6 and 14. As amended to overcome the rejection under 35 U.S.C. 103, claim 6 reads as follows:

> **6. (currently amended)** An improved class loader that, when executed by a host computer system, loads a class in a program execution environment in ~~a~~ the host computer system, the class being required for execution of a program in the program execution environment and the program including a first association between symbolic names in the program and encrypted forms of symbolic names defined in the class and the improved class loader being characterized in that:
>
> ~~the improved class loader extends the class~~ on execution of the program in the program execution environment, the improved class loader ~~by~~

~~using~~ uses the first association and a second association between the encrypted forms and information used to resolve the symbolic names defined in the class to resolve the symbolic names in the program, and

~~adding~~ adds a method to the program which determines whether the program has been modified by the host.

The combination of references must thus disclose a class loader for which the following limitations hold:

- The class loader "loads a class in a program execution environment in the host computer system" The class is "required for execution of a program in a program execution environment";
- the program includes "a first association between symbolic names in the program and encrypted forms of symbolic names defined in the class";
- the class loader "uses the first association and a second association between the encrypted forms and information used to resolve the symbolic names defined in the class to resolve the symbolic names in the program"; and
- the class loader "adds a method to the program which determines whether the program has been modified by the host".

What the combination of references discloses

The combination of the references discloses none of these limitations because both references disclose only techniques that are applied to Java bytecodes prior to the bytecodes being executed in the Java virtual machine. Beginning with Takeuchi, that reference discloses two separate techniques:

- a technique for encrypting Java byte codes prior to downloading them across a network and decrypting the Java byte codes prior to providing the byte codes to the Java virtual machine (FIG. 1, discussed at col. 8, line 49-col. 10, line 10).
- a technique for authenticating downloaded Java byte codes by computing a digest from the Java byte codes prior to downloading them, downloading the digest with the Java byte codes, recomputing the digest from the downloaded byte codes, and comparing the recomputed digest with the downloaded digest to determine

whether the received Java byte codes differed from the Java bytecodes as downloaded. (FIG. 3, discussed at col. 12, line 61-col. 14, line 6)

Because these techniques are applied *prior to* providing the byte codes to the Java virtual machine (JVM), they can have nothing to do with Java class loaders. A Java class loader loads the symbolic names defined in a Java class the first time a byte code in a program being executed in the JVM references a name defined in the class. From the point of view of the Java class loaders in Takeuchi, the decrypted byte code being executed in Takeuchi's JVM is like any other byte code; similarly, it makes no difference to the JVM and its class loaders whether the byte codes it is executing have been authenticated. See in this regard Applicants' [0061] and [0100].

The same is the case with Mahmoud, which discloses that the byte codes for Java classes may be protected against decompilation by running a program called crema on them which obfuscates the names defined in the classes. As expressly pointed out in the middle of page 3 of the reference, "Crema scrambles these symbolic names and makes references to them in the same way so that the JVM can still achieve the correct linking between classes and packages"; again, from the point of view of the class loaders, the obfuscated byte code is like any other byte code.

Neither of the references discloses anything at all about class loaders. As would be expected from that fact, neither of the references discloses the salient limitations of claim 6 or claim 14: Neither the claims' "program including a first association between symbolic names in the program and encrypted forms of symbolic names defined in the class". Nor the claims' class loader which "uses the first association and a second association between the encrypted forms and information used to resolve the symbolic names defined in the class to resolve the symbolic names in the program". Nor the claims' class loader that "adds a method to the program which determines whether the program has been modified by the host". It should be pointed out with regard to the last point that the authentication using a digest employed by Takeuchi *could* "determine whether the program has been modified by the host", but that Takeuchi's technique *does not* involve using the class loader to "add[] a method to the program" to do this.

Because the combination of Takeuchi and Mahmoud discloses *none* of the above-listed limitations of claim 6 or claim 14, Examiner has failed to make the required *prima facie* case for the rejection of the claims under 35 U.S.C. 103 and the rejection is without basis.

Conclusion

Applicants have amended claims 1 and 6 to overcome Examiner's rejection of claims 1-13 under 35 U.S.C. 103, have further amended claim 6 to correct an error therein, and have demonstrated that the claims as amended are fully supported by the Specification as originally filed. Applicants have further demonstrated that the combination of Takeuchi and Mahmoud fails to disclose all of the limitations of claim 6 or of claim 14 and have thereby demonstrated that Examiner's rejection of claims 6-13 under 35 U.S.C. 103 is without basis. Applicants have thereby fulfilled all of the requirements of 37 C.F.R. 1.111(b) and respectfully request that Examiner continue with his examination and allow the claims as amended, as provided by 37 C.F.R. 1.111(a). Applicants are also providing an IDS with a number of references whose publication dates are later than the earliest priority date for Applicants' application and which are consequently not prior art to Applicants' application but which may aid Examiner in his understanding of class loaders and of encryption as applied to Java programs. No fees are believed to be required for this amendment. Should any be, please charge them to deposit account number 501315.

Respectfully submitted,

/Gordon E. Nelson/

Attorney of record,
Gordon E. Nelson
57 Central St., P.O. Box 782
Rowley, MA, 01969,
Registration number 30,093
Voice: (978) 948-7632
Fax: (978) 945-5550
email: genelson@comcast.net

10/7/2009

Date